AN1401
应用笔记

UM321xD ADC 应用手册

版本：V1.0

广芯微电子（广州）股份有限公司

http://www.unicmicro.com/

版本修订

版本	日期	描述
V1.0	2022.06.02	初始版

目录

1 ADC 简介

12 位 ADC 是一种采用逐次逼近方式的模拟数字转换器。ADC 可以转换 7 个外部通道的模拟信号。模拟看门狗允许应用程序来检测输入电压是否超出用户设定的阈值。各种通道的 A/D 转换可以配置成单次转换模式。ADC 转换的结果存储在 12 位数据寄存器中。

可配合《UM321xD_用户手册》和“Driver&Example\ADC”中的代码用例进行应用。

2 ADC 软件配置

2.1 ADC 宏定义

1. **ADC 通道定义：**

```
#define ADC_CH0                                 0
#define ADC_CH1                                 1
#define ADC_CH2                                 2
#define ADC_CH3                                 3
#define ADC_CH5                                 5
#define ADC_CH6                                 6
#define ADC_CH7                                 7
```

从上到下的定义为通道 0、通道 1、通道 2、通道 3、通道 5、通道 6 和通道 7，共 7 个外部通道。

2. **ADC 使能、失能和无状态定义：**

```
#define ADC_ENABLE  1
#define ADC_DISABLE 0
#define ADC_NONE    0
```

从上到下的定义为 ADC 使能、ADC 失能和 ADC 无状态。

3. **ADC DMA 模式定义：**

```
#define ADC_DMA_MODE_1  1
#define ADC_DMA_MODE_2  2
```

从上到下的定义为 DMA 模式 1 和 DMA 模式 2。

4. **ADC RXFIFO 数据可用触发数量定义：**

```
#define ADC_FIFO_1      0
```

定义为 RXFIFO 数据可用触发数量为 1。

5. **ADC 常规序列扫描模式定义：**

```
#define ADC_SINGLE_SCAN         1
```

定义为常规序列扫描模式。

6. **ADC 常规序列转换信号触发边沿定义：**

```
#define ADC_EDGE_UP             1
#define ADC_EDGE_DOWN           2
#define ADC_EDGE_UPORDOWN       3
```

从上到下的定义为上升沿触发、下降沿触发和双边沿触发。

7. ADC 通道平均转换次数定义：

```
#define ADC_AVERAGE_NUMBER_1    0
#define ADC_AVERAGE_NUMBER_2    1
#define ADC_AVERAGE_NUMBER_4    2
#define ADC_AVERAGE_NUMBER_8    3
#define ADC_AVERAGE_NUMBER_16   4
#define ADC_AVERAGE_NUMBER_32   5
#define ADC_AVERAGE_NUMBER_64   6
#define ADC_AVERAGE_NUMBER_128  7
```

从上到下的定义平均转换次数为 1 次、2 次、4 次、8 次、16 次、32 次、64 次和 128 次。

8. ADC 常规序列位号定义：

```
#define ADC_SERIES_NUMBER_1     0
#define ADC_SERIES_NUMBER_2     1
#define ADC_SERIES_NUMBER_3     2
#define ADC_SERIES_NUMBER_4     3
#define ADC_SERIES_NUMBER_5     4
#define ADC_SERIES_NUMBER_6     5
#define ADC_SERIES_NUMBER_7     6
#define ADC_SERIES_NUMBER_8     7
```

从上到下的定义常规序列第 1 位至第 8 位。

9. ADC 常规序列定义：

```
#define ADC_SERIES_REGULAR      1
```

定义为常规序列转换，配置常规序列通道位置数。

10. ADC 模拟看门狗事件触发条件定义：

```
#define ADC_WDG_TRIGGER_OUTSIDE_SCOPE   0
#define ADC_WDG_TRIGGER_WITHIN_SCOPE    1
```

从上到下的定义为转换数据原码超出阈值范围和转换数据原码在阈值范围内。

11. ADC 当前状态定义：

```
#define ADC_STATUS_IDLE_STATE   0
#define ADC_STATUS_CH_CMPLT     1
#define ADC_STATUS_RGL_CMPLT    2
#define ADC_STATUS_CONV_CNT     4
#define ADC_STATUS_CH_NUM       11
#define ADC_STATUS_RGL_NUM      16
#define ADC_STATUS_FIFO_EMPTY   25
#define ADC_STATUS_FIFO_FULL    26
#define ADC_STATUS_WDG_CMPLT    27
```

从上到下的定义为空闲状态标志、通道转换完成脉冲信号状态、常规序列转换完成脉冲信号状态、当前转换位置已完成的转换次数、正在转换的通道编号、常规序列中正在转换的位置编号、RXFIFO 空状态标志、RXFIFO 满状态标志和看门狗比较通道数据完成状态标志。

12. 中断状态定义：

```
#define ADC_INT_STATUS_EORC     0
#define ADC_INT_STATUS_FIFO_OVF 3
#define ADC_INT_STATUS_WDG_CH   4
```

从上到下的定义为常规序列转换结束中断标志、RXFIFO 溢出中断标志和通道数据看门狗报警中断标志。

2.2 单通道单次采样

2.2.1 ADC 初始化流程

ADC 初始化流程如下：

1. 调用 gpio.h 中的 gpio_config_ads 接口，将 PD4（ADC 通道 0）配置为模拟接口。
2. 调用 adc.h 中的 adc_clk_config 接口使能 ADC 时钟。
3. 调用 adc.h 中的 adc_connect_volbuffer_config 接口，断开 ADC 与单位增益电压缓冲器的连接。
4. 调用 adc.h 中的 adc_regular_series_config 接口，其中 convert_mode 传入 ADC_SINGLE_SCAN 宏，设置单次扫描模式。不选择触发信号源，禁止信号触发。
5. 调用 adc.h 中的 adc_clock_control_config 接口，配置分频系数。
6. 调用 adc.h 中的 adc_series_channel_config 接口，设置常规序列在每一位置中的转换通道。例如位置 1 为转换通道 0，位置 2 为转换通道 1…，位置 8 为转换通道 7，可以每一位置或部分位置中为相同的转换通道。并设置转换通道的转换次数。
7. 调用 adc.h 中的 adc_convert_number_config 接口，设置启动转换后，常规序列中的通道转换位置数。
8. 调用 adc.h 中的 adc_controller_config 接口，使能 ADC 控制器。

代码示例如下图所示：

```
/*********************************************************************
 * function    : adc_single_ch_init
 * Description: 单通道初始化
 * input       : uint8_t convert_mode
 * return      : none
 *********************************************************************/
void adc_single_ch_init(uint8_t convert_mode)
{
    gpio_config_ads(GPIOD, PIN4, GPIO_ADS_ANALOG);  //对应 IO 使能为模拟接口

    adc_clk_config(ADC_ENABLE);                     //使能 ADC 时钟

    adc_connect_volbuffer_config(ADC_DISABLE);      //关闭电压增益缓冲器

    adc_regular_series_config(convert_mode, ADC_NONE, ADC_NONE);//配置为单次扫描，不
选择触发信号源，禁止信号触发

    adc_clock_control_config(12);                   //PCLK 进行12分频，即 ADC 采样率为
96/16/12=500KHz

    adc_series_channel_config(ADC_SERIES_NUMBER_1, ADC_CH0,
ADC_AVERAGE_NUMBER_1);//常规位序1为转换通道7，转换1次

    adc_convert_number_config(ADC_SERIES_REGULAR, 1);//常规序列转换位置数1位

    adc_controller_config(ADC_ENABLE);              //使能 ADC 控制器
}
```

图 2-1：ADC 初始化流程代码示例图

2.2.2 ADC 采样流程

ADC 采样流程如下：

1. 调用 adc.h 中的 adc_power_config 接口，开启 ADC 电源。
2. 调用 adc.h 中的 adc_convert_start 接口，启动 ADC 转换（**注意**：开启 ADC 电源后应该立刻启动 ADC 转换）。
3. 调用 adc.h 中的 adc_get_current_status 接口，入口参数为空闲状态宏 ADC_STATUS_IDLE_STATE，等待 ADC 进入空闲状态。
4. 调用 adc.h 中的 adc_get_channel_value 接口，该接口返回值为对应入口参数的通道转换值。
5. 再次调用 adc_power_config 接口，关闭 ADC 电源。
6. 可将读取的转换值通过计算转换为相应的电压值。

代码示例如下图所示：

```
adc_power_config(ADC_ENABLE);                    //启动 ADC 电源
adc_convert_start();                             //启动 adc 转换
while(adc_get_current_status(ADC_STATUS_IDLE_STATE) != 1);//等待转换完成，adc
进入空闲状态
adc_value = adc_get_channel_value(ADC_CH0);      //读取通道转换值
adc_power_config(ADC_DISABLE);                   //关闭 ADC 电源
```

图 2-2：ADC 采样流程代码示例图

2.2.3 单通道单次采样应用

单通道单次采样应用代码示例如下：

```
/*****************************************************************************
 * function   : adc_singlech_singlescan_test
 * Description: 单通道单次采集
 * input : none
 * return: none
 *****************************************************************************/
void adc_singlech_singlescan_test(void)
{
    uint16_t adc_value;
    float vol_value;

    printfS("Start single scan of single channel test \r\n");

    adc_single_ch_init(ADC_SINGLE_SCAN);            //adc 初始化

    while(1)
    {
        adc_power_config(ADC_ENABLE);                   //启动 ADC 电源
        adc_convert_start();                            //启动 adc 转换
        while(adc_get_current_status(ADC_STATUS_IDLE_STATE) != 1);//等待转换完成，adc
进入空闲状态
        adc_value = adc_get_channel_value(ADC_CH0);     //读取通道转换值
        adc_power_config(ADC_DISABLE);                  //关闭 ADC 电源

        if(adc_value != 0x8000)
        {
            vol_value = (float)(adc_value*3.283)/4095;
            printfS("adc value = %d \r\n", adc_value);
            printfS("vol value = %0.3fV \r\n", vol_value);
        }
        else
        {
            printfS("数据无效\r\n");
        }

        delay_ms(500);
    }
}
```

图 2-3：单通道单次采样应用代码示例图

2.3 ADC DMA 传输

2.3.1 ADC 初始化流程

ADC 初始化流程如下：

1. 调用 gpio.h 中的 gpio_config_ads 接口，将 PD4（ADC 通道 0）配置为模拟接口。
2. 调用 adc.h 中的 adc_clk_config 接口使能 ADC 时钟。
3. 调用 adc.h 中的 adc_connect_volbuffer_config 接口，断开 ADC 与单位增益电压缓冲器的连接。
4. 调用 adc.h 中的 adc_regular_series_config 接口，设置单次扫描模式，不选择触发信号源，禁止信号触发。
5. 调用 adc.h 中的 adc_dma_mode_config 接口，配置 DMA 模式 2。
6. 调用 adc.h 中的 adc_clock_control_config 接口，配置分频系数。
7. 调用 adc.h 中的 adc_series_channel_config 接口，设置常规序列在每一位置中的转换通道，例如位置 1 为转换通道 0，位置 2 为转换通道 1…位置 8 为转换通道 7，可以每一位置或部分位置中为相同的转换通道。并设置转换通道的转换次数。
8. 调用 adc.h 中的 adc_convert_number_config 接口，设置启动转换后，常规序列中的通道转换位置数。
9. 调用 adc.h 中的 adc_controller_config 接口，使能 ADC 控制器。

   代码示例如下图所示：


```
/*********************************************************************
 * function   : adc_dma_init
 * Description: dma 功能初始化
 * input : none
 * return: none
 *********************************************************************/
void adc_dma_init(void)
{
    gpio_config_ads(GPIOD, PIN4, GPIO_ADS_ANALOG);  //对应 IO 使能为模拟接口

    adc_clk_config(ADC_ENABLE);                     //使能 ADC 时钟

    adc_connect_volbuffer_config(ADC_DISABLE);      //关闭电压增益缓冲器

    adc_regular_series_config(ADC_SINGLE_SCAN, ADC_NONE, ADC_NONE);//配置为单次扫描，
不选择触发信号源，禁止信号触发

    adc_dma_mode_config(ADC_DMA_MODE_2);            //使能 DMA 模式 2

    adc_clock_control_config(12);                   //PCLK 进行 12 分频，即 ADC 采样率为
96/16/12=500KHz

    adc_series_channel_config(ADC_SERIES_NUMBER_1, ADC_CH0, ADC_AVERAGE_NUMBER_1);//
常规位序 1 为转换通道 7，转换 1 次

    adc_convert_number_config(ADC_SERIES_REGULAR, 1);//常规序列转换位置数 1 位

    adc_controller_config(ADC_ENABLE);              //使能 ADC 控制器

}
```

图 2-4：ADC 初始化流程代码示例图

2.3.2 ADC DMA 传输流程

ADC DMA 传输流程如下：

1. 调用 dma.h 中的 dmac_transfer 接口，配置 DMA 传输通道、源地址、目的地址和外设到内存的传输模式。
2. 调用 adc.h 中的 adc_power_config 接口，开启 ADC 电源。
3. 调用 adc.h 中的 adc_convert_start 接口，启动 ADC 转换（**注意**：开启 ADC 电源后应该立刻启动 ADC 转换）。

4. 调用 adc.h 中的 adc_get_current_status 接口，入口参数为空闲状态宏 ADC_STATUS_IDLE_STATE，等待 ADC 进入空闲状态。
5. 调用 adc.h 中的 adc_get_channel_value 接口，该接口返回值为对应入口参数的通道转换值。
6. 再次调用 adc_power_config 接口，关闭 ADC 电源。
7. 可通过打印目的地址中的转换值和源地址中的转换值，两个值相同即传输成功。

代码示例如下图所示：

```
        dmac_transfer(DMA_Channel_0, (uint32_t)&REG_ADC_CHDAT(ADC_CH0), 0x20001000,
1, TIM02SRAM);//启动 dma 传输
        adc_power_config(ADC_ENABLE);                    //启动 ADC 电源
        adc_convert_start();                           //启动 adc 转换
        while(adc_get_current_status(ADC_STATUS_IDLE_STATE) != 1);//等待转换完成，adc
进入空闲状态
        adc_power_config(ADC_DISABLE);                   //关闭 ADC 电源
        printfS("转换完成\r\n");
```

图 2-5：ADC DMA 传输流程代码示例图

2.3.3 ADC DMA 传输应用

ADC DMA 传输应用代码示例如下所示：

```
/*********************************************************************
 * function   : adc_dma_test
 * Description: adc & dma 功能测试
 * input : none
 * return: none
 *********************************************************************/
void adc_dma_test(void)
{
    printfS("Start adc & dma test \r\n");

    adc_dma_init();                                  //adc 初始化
    dmac_enable();                                   //dma 使能
    dmac_int(DMA_Channel_0, adc_dma_pro, NULL);      //dma 初始化

    *(volatile uint32_t *)(0x20001000) = (uint32_t)0x00;//SRAM 待传输地址内容清 0

    while(1)
    {
        dmac_transfer(DMA_Channel_0, (uint32_t)&REG_ADC_CHDAT(ADC_CH0), 0x20001000,
1, TIM02SRAM);//启动 dma 传输
        adc_power_config(ADC_ENABLE);                //启动 ADC 电源
        adc_convert_start();                         //启动 adc 转换
        while(adc_get_current_status(ADC_STATUS_IDLE_STATE) != 1);//等待转换完成，adc
进入空闲状态
        adc_power_config(ADC_DISABLE);               //关闭 ADC 电源
        printfS("转换完成\r\n");
        delay_ms(1000);
    }
}
```

图 2-6：ADC DMA 传输应用代码示例图

2.4 模拟看门狗

2.4.1 ADC 初始化流程

ADC 初始化流程如下：

1. 调用 gpio.h 中的 gpio_config_ads 接口，将 PD4（ADC 通道 0）配置为模拟接口。
2. 调用 adc.h 中的 adc_clk_config 接口使能 ADC 时钟。

3. 调用 adc.h 中的 adc_connect_volbuffer_config 接口，断开 ADC 与单位增益电压缓冲器的连接。
4. 调用 adc.h 中的 adc_regular_series_config 接口，设置单次扫描模式，不选择触发信号源，禁止信号触发。
5. 调用 adc.h 中的 adc_clock_control_config 接口，配置分频系数。
6. 调用 adc.h 中的 adc_series_channel_config 接口，设置常规序列在每一位置中的转换通道，例如位置 1 为转换通道 0，位置 2 为转换通道 1…位置 8 为转换通道 7，可以每一位置或部分位置中为相同的转换通道。并设置转换通道的转换次数。
7. 调用 adc.h 中的 adc_convert_number_config 接口，设置启动转换后，常规序列中的通道转换位置数。
8. 调用 adc.h 中的 adc_wdg_threshold_config 接口，设置模拟看门狗的上下阈值电压。
9. 调用 adc.h 中的 adc_wdg_trigger_config 接口，设置为采样电压在上下限范围外触发报警。
10. 调用 adc.h 中的 adc_wdg_channel_config 接口，设置 ADC 通道 0 为模拟看门狗通道。
11. 调用 adc.h 中的 adc_controller_config 接口，使能 ADC 控制器。
    代码示例如下图所示：


```
/*********************************************************************
 * function   : adc_wdg_init
 * Description: 模拟看门狗初始化
 * input : none
 * return: none
 *********************************************************************/
void adc_wdg_init(void)
{
    gpio_config_ads(GPIOD, PIN4, GPIO_ADS_ANALOG);  //对应 IO 使能为模拟接口

    adc_clk_config(ADC_ENABLE);                     //使能 ADC 时钟

    adc_connect_volbuffer_config(ADC_DISABLE);      //关闭电压增益缓冲器

    adc_regular_series_config(ADC_SINGLE_SCAN, ADC_NONE, ADC_NONE);//配置为单次扫描，
不选择触发信号源，禁止信号触发

    adc_clock_control_config(12);                   //PCLK 进行 12 分频，即 ADC 采样率为
96/16/12=500KHz

    adc_series_channel_config(ADC_SERIES_NUMBER_1, ADC_CH0, ADC_AVERAGE_NUMBER_1);//
常规位序 1 为转换通道 0，转换 1 次

    adc_convert_number_config(ADC_SERIES_REGULAR, 1);//常规序列转换位置数 1 位

    adc_wdg_threshold_config(2.5, 1.0);             //设置模拟看门狗上下限电压
    adc_wdg_trigger_config(ADC_WDG_TRIGGER_OUTSIDE_SCOPE);//采样电压在上下限范围外触发
报警
    adc_wdg_channel_config(ADC_CH0, ADC_ENABLE);    //使能通道 0 模拟看门狗

    adc_controller_config(ADC_ENABLE);              //使能 ADC 控制器

}
```

图 2-7：ADC 初始化流程代码示例图

2.4.2 ADC 中断配置

ADC 中断配置如下：

- 调用 adc.h 中的 adc_irq_config 接口，使能 ADC 通道数据看门狗报警中断，注册模拟看门狗回调函数。代码示例如下所示：

```
adc_irq_config(ADC_INT_STATUS_WDG_CH, ADC_ENABLE,adc_wdg_pro);//使能模拟看门狗报
警中断，注册回调函数
```

- 模拟看门狗回调函数中的具体实现由用户自己定义。代码示例如下所示：

```
/*****************************************************************************
 * function   : adc_wdg_pro
 * Description: 模拟看门狗回调函数
 * input : none
 * return: none
 *****************************************************************************/
void adc_wdg_pro(void)
{
    printfS("模拟看门狗中断\r\n");
}
```

2.4.3 ADC 采样流程

ADC 采样流程如下：

1. 调用 adc.h 中的 adc_power_config 接口，开启 ADC 电源。
2. 调用 adc.h 中的 adc_convert_start 接口，启动 ADC 转换（**注意**：开启 ADC 电源后应该立刻启动 ADC 转换）。
3. 调用 adc.h 中的 adc_get_current_status 接口，入口参数为空闲状态宏 ADC_STATUS_IDLE_STATE，等待 ADC 进入空闲状态。
4. 调用 adc.h 中的 adc_get_channel_value 接口，该接口返回值为对应入口参数的通道转换值。
5. 再次调用 adc_power_config 接口，关闭 ADC 电源。
6. 可将读取的转换值通过计算转换为相应的电压值。

代码示例如下图所示：

```
        adc_power_config(ADC_ENABLE);                    //启动 ADC 电源
        adc_convert_start();                             //启动 adc 转换
        while(adc_get_current_status(ADC_STATUS_IDLE_STATE) != 1);//等待转换完成，adc
进入空闲状态
        adc_value = adc_get_channel_value(ADC_CH0);      //读取通道转换值
        adc_power_config(ADC_DISABLE);                   //关闭 ADC 电源
```

图 2-8：ADC 采样流程代码示例图

2.4.4 模拟看门狗应用

模拟看门狗应用代码示例如下图所示：

```
/*****************************************************************************
 * function   : adc_wdg_test
 * Description: 模拟看门狗
 * input : none
 * return: none
 *****************************************************************************/
void adc_wdg_test(void)
{
    uint16_t adc_value;
    float vol_value;

    printfS("Start wdg test \r\n");

    adc_wdg_init();                                    //adc 初始化

    adc_irq_config(ADC_INT_STATUS_WDG_CH, ADC_ENABLE,adc_wdg_pro);//使能模拟看门狗报警中断，注册回调函数

    while(1)
    {
        adc_power_config(ADC_ENABLE);                  //启动 ADC 电源
        adc_convert_start();                           //启动 adc 转换
        while(adc_get_current_status(ADC_STATUS_IDLE_STATE) != 1);//等待转换完成，adc 进入空闲状态
        adc_value = adc_get_channel_value(ADC_CH0);    //读取通道转换值
        adc_power_config(ADC_DISABLE);                 //关闭 ADC 电源

        if(adc_value != 0x8000)
        {
            vol_value = (float)(adc_value*3.283)/4095;
            printfS("adc value = %d \r\n", adc_value);
            printfS("vol value = %0.3fV \r\n", vol_value);
        }
        else
        {
            printfS("数据无效\r\n");
        }
        delay_ms(300);
    }
}
```

图 2-9：模拟看门狗应用代码示例图